Python est un langage de programmation créé par *Guido Van rossum* en 1991. Contrairement à **Scratch**, vous devez taper le code que vous voulez exécuter. Les **instructions** en anglais sont à connaître: on appelle cela la **syntaxe**. Ce code se tape dans un environnement de travail où il sera **interprété**: **chaque instruction est traduite et la traduction est exécutée dans la foulée.**

Ada Lovelace née en 1815 est la première codeuse de l'humanité.

0.1 Les variables

Une variable est l'association entre un nom et une valeur. L'affectation est une instruction qui permet de donner une valeur à une variable. Par exemple, vous écrivez $A \longleftarrow 5$ en pseudo-code (type BAC). Python utilise le symbole = pour cela: `A=5`. Cela n'a donc rien à voir avec l'égalité au sens mathématique.

En bref Une variable possède un nom ou *identificateur* qui fait référence à un emplacement de la mémoire de l'ordinateur contenant un objet. Toute variable est *in fine* représentée pour le processeur par des octets, et c'est grâce au type de cette variable que le processeur saura si ces octets représentent un entier, un réel, une chaîne de caractères etc. Python est un langage typé **dynamiquement**, c'est-à-dire qu'on peut changer le type d'une variable. A contrario, un langage typé **statiquement** comme C/C++ ou Java force à définir le type des variables et à le conserver au cours de la vie de la variable.

Cette facilité de Python peut également être source de confusion et il convient d'avoir bien réfléchi au type des données que l'on utilise. Nous y reviendrons au moyen des annotations.

Syntaxe dans différents langages

En **C et C++**	En **Python**	En **Javascript**
int a = 2	a = 2	**var** a = 2
float b = 3.14	b = 3.14	**var** b = 3.14

- Le symbole `=` permet d'effectuer une *affectation* et non un test d'égalité.
- En C/C++, le type de la variable a du être déclarée: b est de type `float` (nombre à virgule) et a de type `int` (nombre entier). Ce type ne pourra plus être changé.
- En Python et Javascript, le type des variables est déterminé lors de l'interprétation et pourra éventuellement être changé. En Javascript, le mot clé `var` permet de déclarer une variable globale, `let` permet de déclarer une variable dont la portée est limitée à un bloc: on dit qu'elle est **locale** à ce bloc.

On dit que C est un langage à **typage statique**, alors que Python et Javascript sont à **typage dynamique**. C/C++ sont des langages compilés: l'ensemble du programme est traduit en une fois et le résultat est mis dans un ou plusieurs fichier binaires contenant le programme en langage machine.

0.2 Séquence

En bref La séquence est une suite d'instruction. Dans ce **bloc**, les instructions sont exécutées les unes après les autres. Les instructions peuvent contenir des expressions. Une expression est une suite de caractères définissant une valeur. Pour connaître cette valeur, la machine doit évaluer l'expression.

CC-BY-SA

Exemple: `10//3` est une expression, elle est évaluée à 3.

0.2.1 Boucle

Un ordinateur est plutôt doué pour le travail répétitif: les *boucles* sont faites pour répéter les séquences d'instruction. La notion de bloc d'instruction devient indispensable: il est nécessaire de marquer le bloc d'instructions à répéter. Cela se fait avec l'indentation où on décale le code en Python et les accolades en C/C++.

0.2.2 Boucle bornée

Lorsqu'on connaît le nombre de répétition à effectuer, la boucle est dite bornée. On utilise une boucle `pour`.

Exemple en Python

```
for i in range (1, 3):
    print(i)
print('Vive la NSI')
```

```
>>> %Run Exemple.py
0
1
2
Vive la NSI
```

Le compteur de boucle `i` est un entier, par défaut le pas est de 1.

En bref Les syntaxes en **C** et **Javascript** sont identiques pour la boucle **Pour**. Les blocs d'instructions sont placés entre accolades en C et Javascript. En Python, on utilise deux points `:` suivis d'un décalage du code appelé **indentation**. Cette indentation est obligatoire en Python pour marquer le bloc à répéter. Elle est conseillée dans les autres langages.

Syntaxe dans différents langages

En **C**	En **Python**	En **Javascript**

for (i=0; i < 11; i++)
{
printf(i);
}
	for i **in range**(1,11)**:**
	print(i**2)
		for (i=0; i < 11; i++)
		{
		document.write(i*i);
		}

0.2.3 Boucle non bornée

Lorsqu'on ne connait pas le nombre de tour, on peut répéter un bloc d'instructions tant qu'une *condition* est vérifiée: c'est la boucle **Tant que**

En **C** et **Javascript**	En **Python**

while (a⩽b)
{
a = a+1;
}
	while a⩽b**:**
	a = a+1

Une boucle *Tant que* est susceptible de ne pas s'arrêter, de **diverger**. Une technique pour prouver que cette boucle termine est celle du **variant de boucle.**

Variant de boucle Quantité entière, positive dont dépend l'arrêt de la boucle et qui décroit vers zéro. Dans l'exemple ci-dessus, la quantité **b-a** est un variant de boucle, ce qui assure sa **terminaison**.

0.3 Les fonctions

On peut définir dans tous les langages de programmation des fonctions, celles-ci peuvent être appelées à chaque fois qu'une même séquence d'instruction est nécessaire.

En voici la syntaxe Python, les lignes précédées d'un # sont des commentaires:

```
#Définition de la fonction
def hello_world():
    print("Bonjour à tous !")
#Appel de la fonction
hello_world()
```

Le mot clé `def` sert à définir la fonction dont le nom est `hello_world`. Celle-ci est appelée pour être exécutée par `hello_world()`.

 : Il est essentiel de distinguer le début et la fin du bloc d'instructions:

- `print("Bonjour le monde !")` est une instruction incluse dans le **corps de la fonction**.
- L'appel de la fonction `hello_world()`ne l'est pas: cette instruction est en dehors du **corps de la fonction**.

Le décalage du code appelé **indentation** permet en Python de distinguer les limites de bloc. Pour utiliser des valeurs de variables différentes à chaque appel, il nous faudra utiliser un ou des **paramètres**. Un paramètre se comporte comme une variable locale à la fonction et sa valeur sera donnée à chaque appel par l'utilisateur. Une fonction peut prendre un ou plusieurs paramètres que l'on écrit entre les parenthèses. On passe les valeurs (appelées arguments) au moment de l'appel de la fonction, en voici deux exemples en Python et Javascript:

```
def double(x):
    resultat = 2*x
    return resultat

>>>double(5)
10
```

```
function prix(ht, tva):
{
    var ttc = ht* (1+tva/100);
    return ttc;
}
```

5 est l'argument ou paramètre effectif. Noter l'utilisation du mot clé `return` dans le corps de la fonction `double`.

En bref Les **fonctions** sont des blocs d'instructions que l'ont a nommés: on peut les appeler dans le programme et ainsi **factoriser le code**. Ce bloc appelé **corps de la fonction** est délimité par différentes méthodes selon les langages: des accolades en C++ ou Javascript, l'indentation en Python.

Les fonctions peuvent avoir un ou plusieurs **paramètres** qui permettent de transmettre des valeurs à ce bloc: ces paramètres se comportent alors comme des variables connues seulement dans le corps de la fonction. De même, les variables déclarées dans le corps d'un fonction sont dites **locales** : elles ne sont connues que dans cette fonction. On dit que la **portée de la variable** est limitée à cette fonction.

Le mot clé `return` permet de renvoyer une valeur: celle-ci pourra être utilisée ailleurs dans le programme, contrairement à une variable locale.

```
>>>x = double(5)
>>>x
10
```

```
>>>x = hello_world()
>>>x
None
```

Si des conditions portent sur les paramètres (préconditions) ou la valeur de retour (postconditions), elles doivent être documentées dans le corps de la fonction. On dit que l'on donne sa **spécification**.

Sa **signature** est la donnée de son nom, du type de la valeur qui sera retournée et de ses paramètres.

Il est de plus nécessaire de tester le bon fonctionnement d'une fonction en prévoyant des **jeux de test**.

CC-BY-SA

Syntaxe dans différents langages

En **C et C++**	En **Python**	En **Javascript**

#include <cmath>
float hypotenuse(**int** a, **int** b)
{
  return sqrt(a*a+b*b);
}
	from math **import** sqrt
	def hypotenuse(a, b)**:**
	  return sqrt(a*a+b*b)
		function hypotenuse(a, b)
		{
		  return Math.sqrt(a*a+b*b);
		}

Notez l'import des modules de math en C et Python.

0.4 Structures conditionnelles

Pour tester une condition on utilise l'instruction `if`, éventuellement suivi d'un `else`:

En **C** et **Javascript**	En **Python**

if (a==b)
{
  a = b+1;
}
else
{
  a = a+1;
}
	if a ==b :
	
	  a = b+1
	else:
	  a = a+1;

Pour plus de deux tests, voir `http://maths-code.fr/NSI/1ere/Cours-Python_2020.pdf`.

0.4.1 Instructions basiques

Pseudo-code		PYTHON
Afficher A		**print**(A)
$A \longleftarrow 3$		$A = 3$

Pour i **allant** de 1 à 10
  Bloc d'instructions
Fin Pour
		for i **in range** (1,11)**:**
		  `Bloc d'instructions`

Si *condition 1*
Alors *Action 1*
Si *condition 2*
Alors *Action 2*
Sinon *Action 3*
Fin Si
		if *condition***:**
		  `Bloc d'instructions`
		elif *condition***:**
		  `Bloc d'instructions`
		else:
		  `Bloc d'instructions`

Tant que *condition*
  Action
Fin Tant que
		while *condition***:**
		  `Bloc d'instructions`

Définition d'une fonction f
de paramètres x, y
qui renvoie z
		def `fonction(x, y):`
		  `Bloc d'instructions`
		  return z

N entier aléatoire entre 1 et 6		**import** `random` **as** `r` `N = r.randint(1,6)`
Test $A \neq B$		A != B
Test $A = B$		A == B
Quotient entier de A par B		A // B
x^y		$x**y$
Reste de la division de a par b		$a \% b$